Chapter 1

Introduction

1.1 What is embedded system?

Embedded systems are stand alone computing devices or centers usually dedicated to performing limited computing functions reliably, securely and with minimum upkeep costs[1]. An embedded product uses a microprocessor (or microcontroller) todo one task and one task only[2] . Typically, an embedded system is housed on a single microprocessor board with the programs stored in ROM[3]. The key characteristic, however, is being dedicated to handle a particular task. Since the embedded system is dedicated to specific tasks, design engineers can optimize it to reduce the size and cost of the product and increase the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale.[4]

1.2 Why to choose embedded system?[5]

1. Energy Efficiency (draw minimum power for the purpose)
2. Custom voltage/power requirements (VDC: 12, 14, 24, 72...)
3. Security (hacker proof)
4. Reliability (work without failure for years)
5. Environment (broad temperature range, sealed from chemicals, radiation)

6. Efficient Interaction with user (fewer buttons touch)

7. Integrate with design

1.3 Embedded system in our project

As our project titled “Embedded system based robots motion control” so it’s the major part of our whole project. As the representation of embedded system we have used a microcontroller in our project, along with various other sophisticated devices on board. We’ve controlled the normal motion, reverse motion, back spin of two DC motors with our system, for controlling a typical security robotic car.

Chapter 2

Thinking & Design

In the area where the presence of any human is vulnerable , a Device could survive very easily there. Like in the Battle field, the presence of human guard, who is only employed there for keeping eye and give the information if there is something suspicious. The supply of information would be stopped if the guard will be murdered or injured, but a device would keep doing its work by sending real time video, audio or some other data. By thinking this problem we took steps to making something which could be used as a superlative alternative of human in the vulnerable areas.

2.1 Design

When we think of any Engineering project it's our first and foremost duty to think of a flowchart based block diagram. Lets have a look into our block diagram:

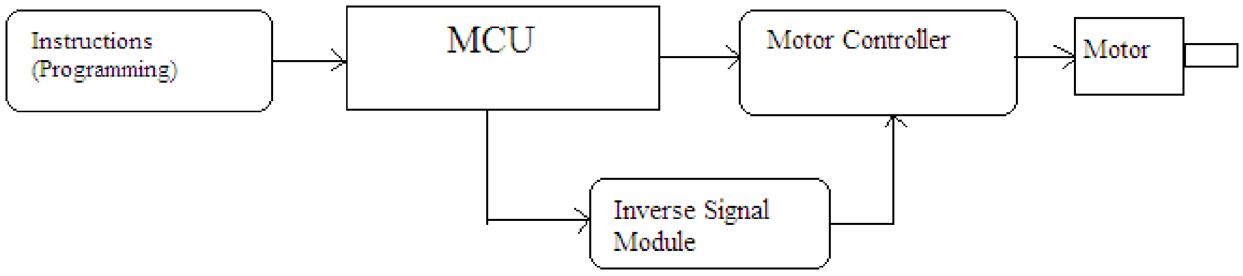


Fig 2.1 : Block Diagram of our Design

2.2 Brief description of Block Diagram

2.21 **Instructions**

The instructions are the reflections of our thinking, that means, the function we want to perform, have to be converted into special programming instructions for microcontroller, as it cannot understand normal human language. As our project is to understanding the embedded system in a better means, we've tried to limit our instructions by not lengthening too much.

2.22 **Microcontroller**

It's the brain of our whole project, it's performs the instructions given, and orders the device connected with it to perform the legitimate functions of the respective device. Actually our project becomes an embedded system based project only for this microcontroller, if we think of using different gate logic IC's etc. then it would not be an embedded system. From the definition of embedded system, we should understand the legitimacy of using a microcontroller.

2.23 **Inverse Signal module**

This part performs the inversion of a signal given into the input of the device, as we are controlling motor's speed, spin and rotation, so it's a must need for us to have a inverse signal of a respective positive signal.

2.24 **Motor Controller**

We’ve think of using a Motor controller as we have to control two motors in different direction, different speed, and spin. We shall discuss the brief of motor controller in the device mounting section.

Chapter 3

Mounting the Device

3.1 The Schematic Diagram

This Schematic has been designed and simulated by Proteus professional edition ISIS design tool.

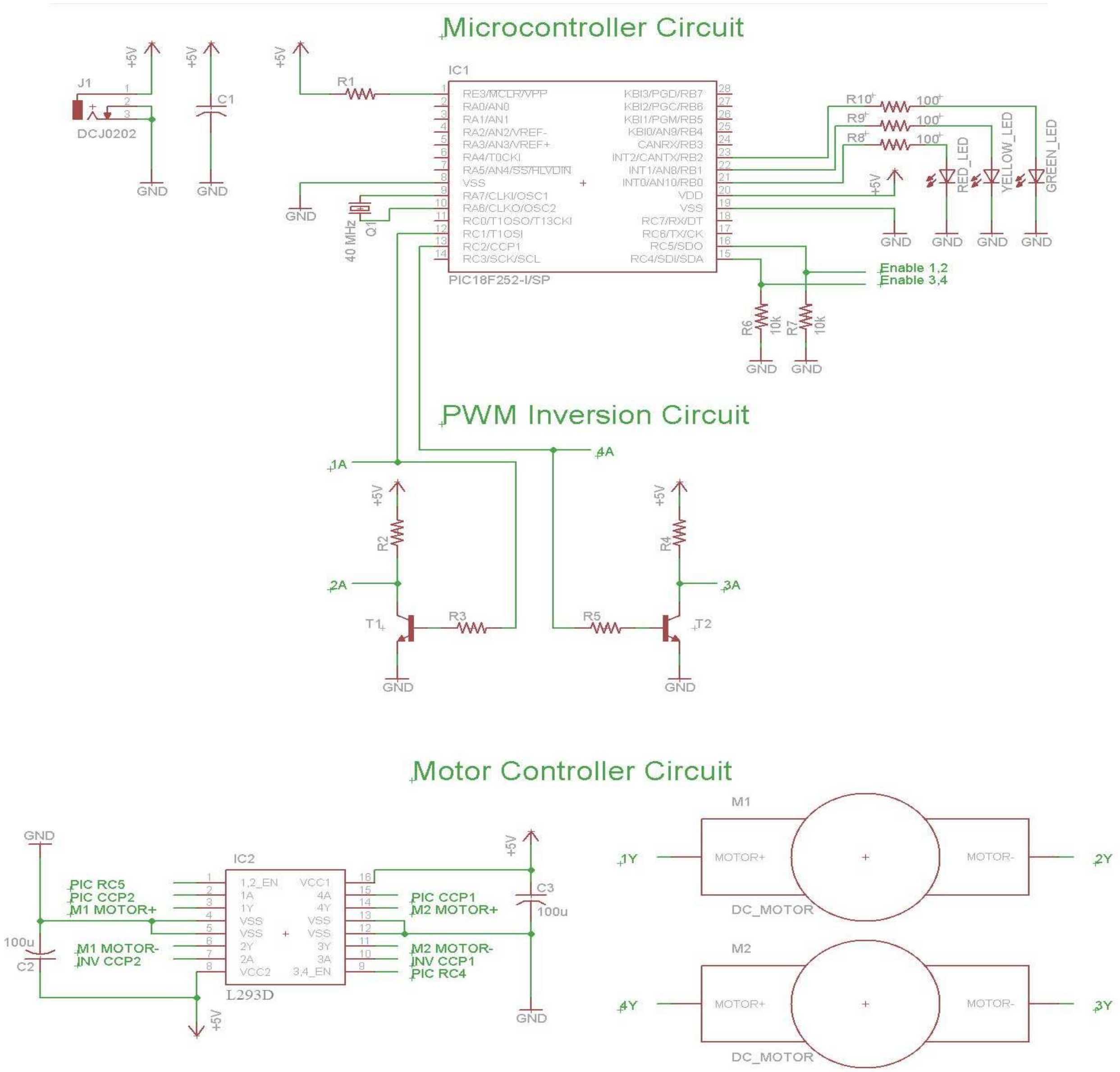


Fig 3.1: The Schematic Diagram

3.2 Parts List

We’ve used the following parts for mounting the device:

1. PIC 18F252 Microcontroller (1 nos)
2. L293D motor driver (1 nos)
3. 2N2222A transistor (2 nos)
4. 40 MHz crystal
5. 10KΩ resistor (7 nos)
6. 100Ω resistor (3 nos)
7. 100µF 25 V capacitor (2 nos)
8. 10µF 25 V capacitor (1 nos)
9. Connecting wire
10. Bread Board
11. Jumper
12. LED (3 nos) Red, Green, Yellow.

3.3 Brief Description of Parts

3.31 The Microcontroller

As we mentioned earlier the microcontroller is the brain of this project, so it’s the most important part of this project. We’ve used the PIC 18F252 Microcontroller.

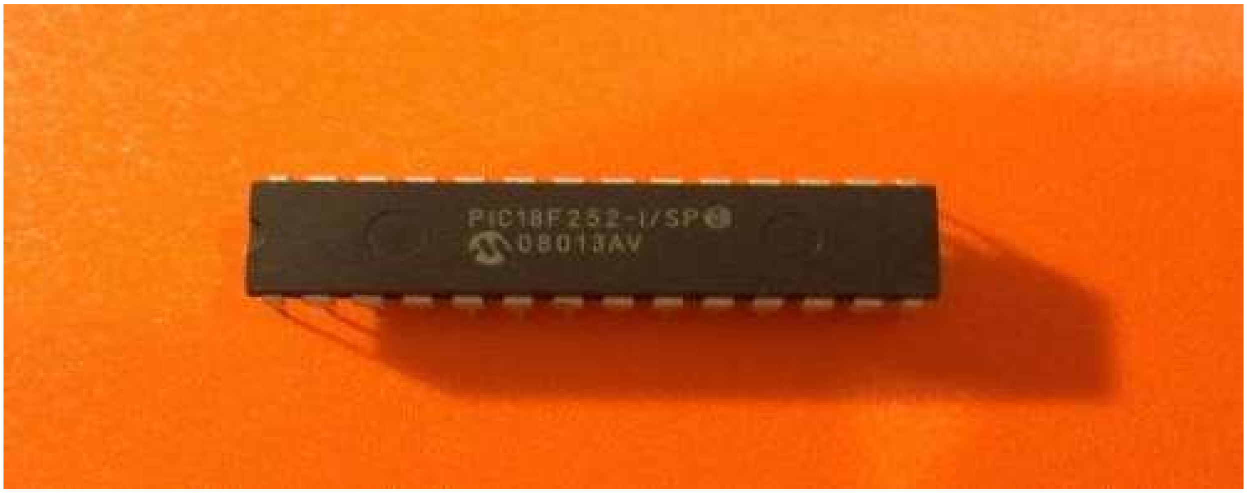


Fig 3.31: A PIC 18F252 microcontroller [6]

Features of PIC18F252 microcontroller[7]:

1. C compiler optimized architecture
2. Linear program memory addressing to 32 Kbytes
3. Linear data memory addressing to 1.5 Kbytes (1536 Bytes)
4. EEPROM 256 Bytes
5. Up to 10 MIPs operation
6. 40 MHz DC clock/oscillator input
7. 4 MHz - 10 MHz oscillator/clock input with PLL (active
8. 16-bit wide instructions, 8-bit wide data path
9. Priority levels for interrupts
10. 8 x 8 Single Cycle Hardware Multiplier

11. High current sink/source 25 mA/25 mA
12. Three external interrupt pins
13. 10-bit ADC, 10 channels, 100K samples per second

3.32 The Block Diagram of PIC18F252

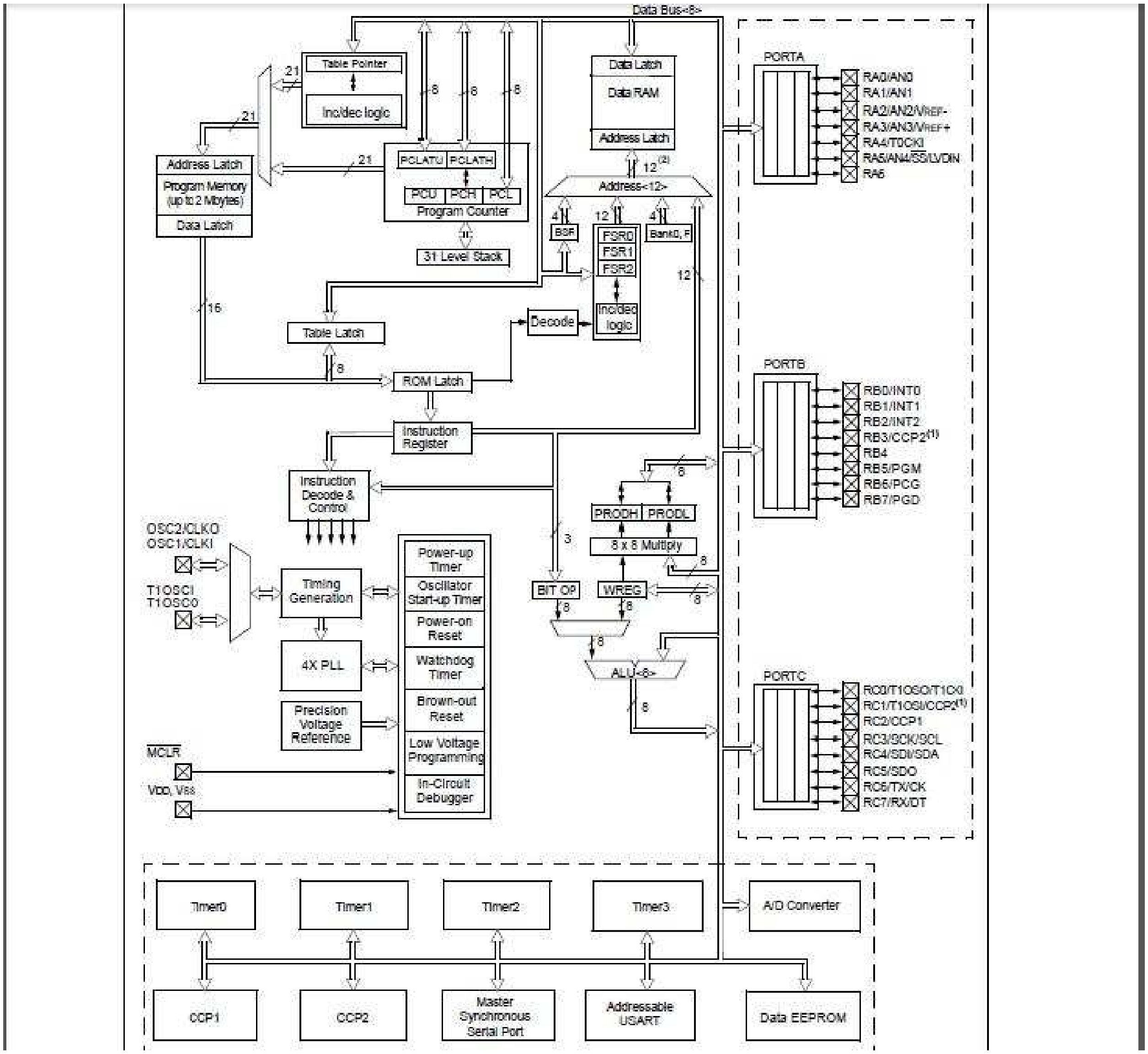


Fig 3.32: The Block Diagram of PIC18F252 [8]

3.33The Pin out of Microcontroller

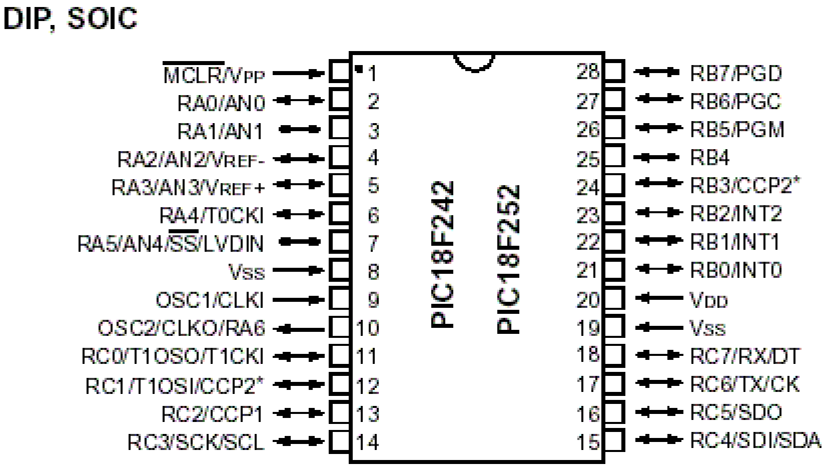


Fig: 3.33: Pin diagram of PIC 18F252 [9]

Pin outs details:

The PIC 18F252 is a 28 pin DIP package, with 3 types of I/O ports namely A,B,C. Port A got 7 pins, port B got 8 pins and port C got 8 pins.

Let's have quick look into the pins and their functions:

Pin1 (MCLR): Master Clear (Reset) input. This pin is an activelow RESET to the device.

Pin2 (RA0): Digital I/O port

Pin3 (RA1): Digital I/O port

Pin4 (RA2): Digital I/O port

Pin5 (RA3): Digital I/O port

Pin6 (RA4): Digital I/O port

Pin7 (RA5): Digital I/O port
Pin8 (Vss): Ground Reference for logic and I/O ports

Pin9 (OSC1/CLK1): Oscillator crystal input or external clock source input.

ST buffer when configured in RC mode, CMOS otherwise.

Pin10 (OSC2/CLK0): Oscillator crystal output. Connects to crystal or

Resonator in Crystal Oscillator mode

Pin11 (RC0): Digital I/O port

Pin12 (RC1): Digital I/O port

Pin13 (RC2): Digital I/O port

Pin14 (RC3): Digital I/O port

Pin15 (RC4): Digital I/O port

Pin16 (RC5): Digital I/O port

Pin17 (RC6): Digital I/O port

Pin18 (RC7): Digital I/O port

Pin19 (Vss): Ground Reference for logic and I/O ports

Pin20 (V_{DD}): Positive Supply for logic and I/O ports

Pin21 (RB0): Digital I/O port

Pin22 (RB1): Digital I/O port

Pin23 (Rb2): Digital I/O port

Pin24 (RB3): Digital I/O port

Pin25 (RB4): Digital I/O port

Pin26 (RB5): Digital I/O port

Pin27 (RB6): Digital I/O port

Pin28 (RB7): Digital I/O port

Functioning of PIC18F252:

When we want to connect the PIC18F252 we should check whether the connection Properly done for functioning. We have to put a crystal between pin no. 9 and 10. Pin no 8&19 should be grounded where no 20 have to be connected to the Vdd pin. Pin no. 1 should be connected to Vdd by a resistor as the pin is active low.

3.34 The Motor Driver:

We have used L293D motor driver for our project. It's a push pull four channel driver with diode. The Device is a monolithic integrated high voltage, High current four channel driver designed to Accept standard DTL or TTL logic levels and drive inductive loads (such as relays solenoids, DC and stepping motors) and switching power transistors[10]. It's a quadruple half H-bridge. What this means is that it can control two DC motor or a stepper motor, this is a cheap and reliable IC that fits nicely to the bread board we've used for this project To simplify use as two bridges each pair of channels Is equipped with an enable input. A separate Supply input is provided for the logic, allowing operation At a lower voltage and internal clamp diodesare included. This device is suitable for use in switching applications at frequencies up to 5 kHz.

Fig 3.3.1: A L293D Motor Driver

Features of L293D[11]

1. 600mA OUTPUT CURRENT CAPABILITY
PER CHANNEL

2. 1.2A PEAK OUTPUT CURRENT (non repetitive)
PER CHANNEL

3. ENABLE FACILITY

4. OVERTEMPERATURE PROTECTION

5. LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V
(HIGH NOISE IMMUNITY)

6. INTERNAL CLAMP DIODES

3.35 Pin-outs of L293D motor driver

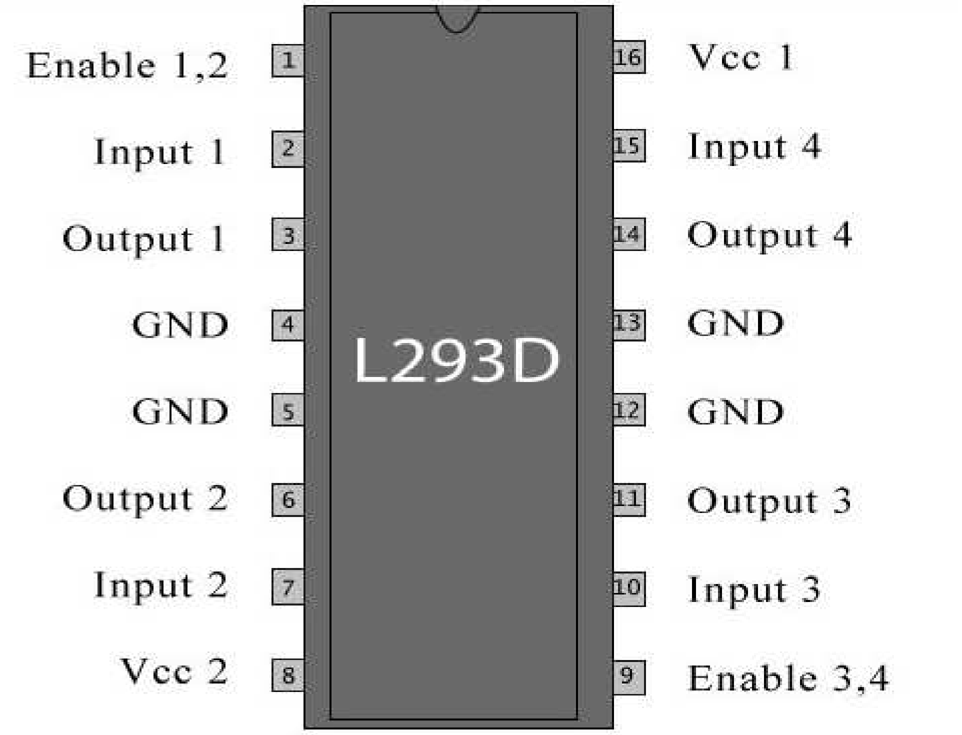


Fig 3.35: Pin diagram of L293D motor driver[12]

Brief Functions of Pins

Pin1 (Enable 1,2): Enable pin for I/O ports (1 &2)

Pin2 (Input1): Digital Input pin, usually a PWM pulse.

Pin3 (Output1): Digital output pin, used to take the output for connecting the motors

Pin4 (Gnd): Ground reference for I/O ports

Pin5 (Gnd): Ground reference for I/O ports

Pin6 (Output2): Digital output pin, used to take the output for connecting the motors

Pin7 (Input2): Digital input pin, usually a PWM pulse.

Pin8 (Vcc2): positive voltage supply pin for I/O ports.

Pin9 (Enable 3, 4): Enable pin for I/O ports (3 &4)

Pin10 (Input3): Digital input pin, usually a PWM pulse.

Pin 11 (Output3): Digital output pin, used to take the output for connecting the motors

Pin12 (Gnd): Ground reference for I/O ports

Pin13 (Gnd): Ground reference for I/O ports

Pin14 (Output4): Digital output pin, used to take the output for connecting the motors

Pin15 (Input4): Digital input pin, usually a PWM pulse.

Pin16 (Vcc1): positive voltage supply pin for I/O ports.

Functioning of L293D motor driver:

For proper functioning of motor driver, the connection of different pins must be set appropriately. Pin no.1(enable1,2) and pin no.9(enable3,4) must be connected to Vcc, where pin no. 4&5 and 12 &13 must be grounded.

Lets have look to the proper connection diagram of a L293D motor driver:

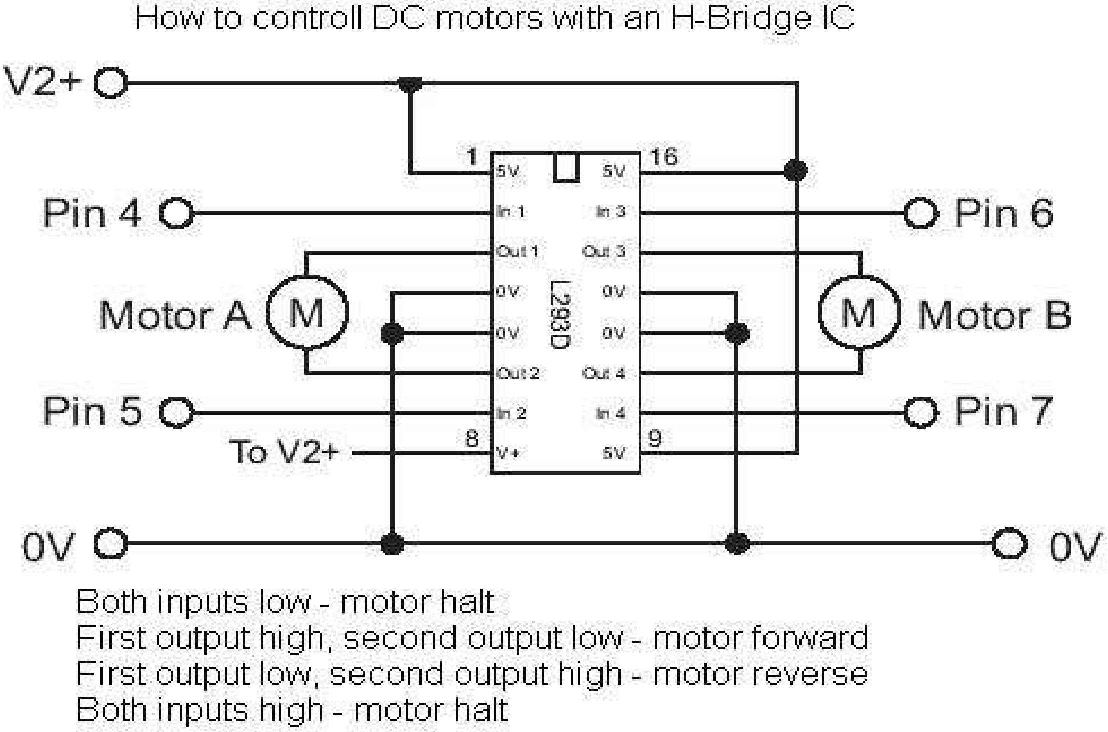

How to controll DC motors with an H-Bridge IC
V2+
Pin 4
Motor A
M
Pin 5
To V2+
0V
1
5V
In 1
Out 1
0V
0V
Out 2
In 2
8
V+
L293D
16
5V
In 3
Out 3
0V
0V
Out 4
In 4
9
5V
Pin 6
M
Motor B
Pin 7
0V
Both inputs low - motor halt
First output high, second output low - motor forward
First output low, second output high - motor reverse
Both inputs high - motor halt

Fig 3.3.2: Connection Diagram for L293D motor controller[13]

3.36: The Transistor

The transistor we used in our project is 2N2222A, which is a NPN bipolar switching transistor. It's a icon planar epitaxial NPN transistor in Jedec TO-18 metal case. It is designed for high speed switching application at collector current up to500mA, and feature useful current gain over a wide range of collector current, low leakage currents and low saturation voltage[14]. In our project this generic transistors will be used in assist the PIC microcontroller for sending the proper signal to the motor controller. The motor controller requires differential PWM input, and the transistor will create this differential input for each motor.

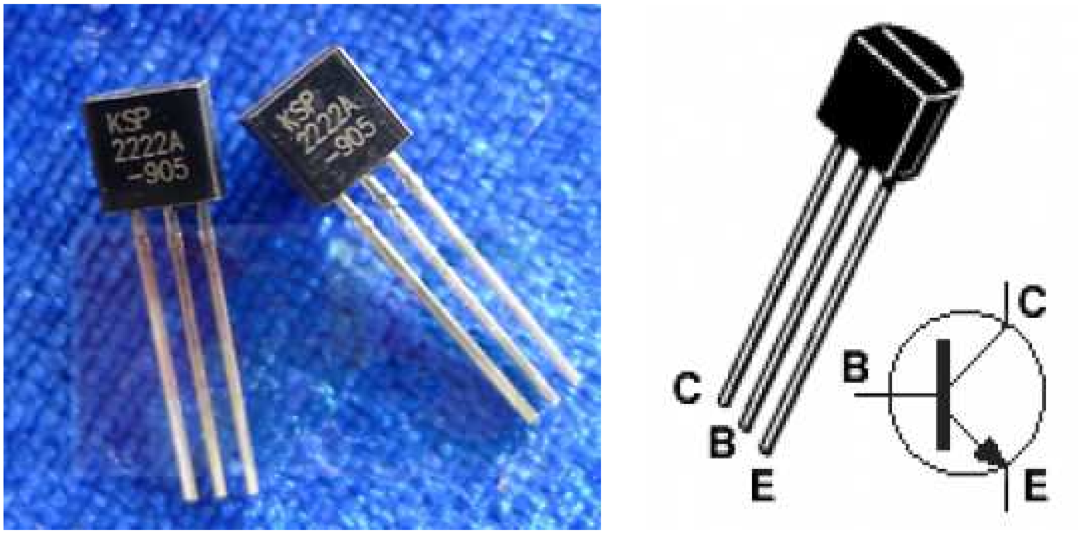


Fig 3.36: The 2N222A transistor[15]

Features of 2N2222A transistor [16]

1. Low Collector Base voltage 75V
2. Maximum current 800mA
3. Fast switching 335nS
4. Linear amplification and switching
5. Power dissipation 500mW

3.37:Absolute Maximum rating of 2N2222A[17]

Symbol	**Parameter**	**Value**	**Units**
V_{CEO}	Collector-Emitter Voltage	40	V
V_{CBO}	Collector-Base Voltage	75	V
V_{EBO}	Emitter-Base Voltage	6	V
I_C	Collector Current	1	A
T_{STG}	Operating Temperature	-55~150	ºC
P_{TOT}	Power Dissipation	0.5	mW

3.38 Crystal Oscillator

A **crystal oscillator** is an electronic oscillator circuit that uses the mechanical resonance of a vibrating crystal of piezoelectric material to create an electrical signal with a very precise frequency. This frequency is commonly used to keep track of time (as in quartz wristwatches), to provide a stable clock signal for digital integrated circuits, and to stabilize frequencies for radio transmitters and receivers. The most common type of piezoelectric resonator used is the quartz crystal, so oscillator circuits incorporating them became known as crystal oscillators, but other piezoelectric materials including polycrystalline ceramics are used in similar circuits[18]

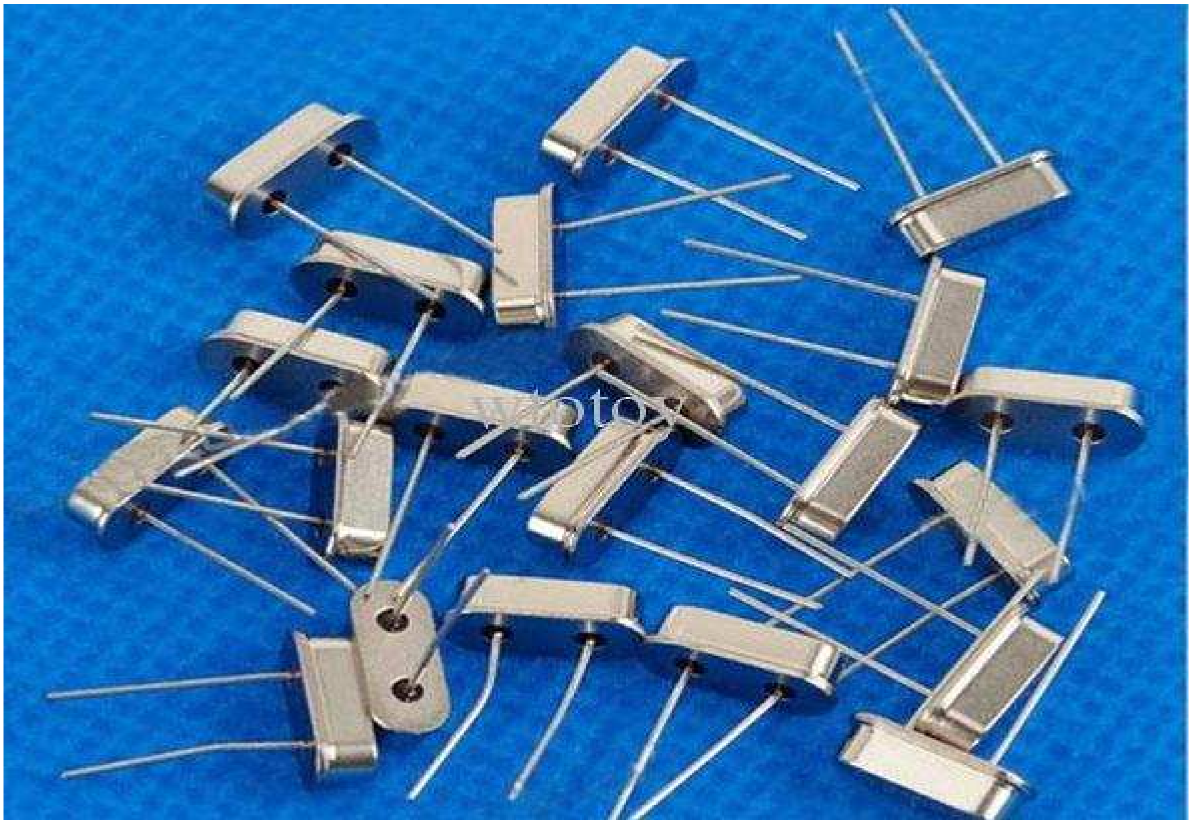


Fig 3.38: Crystal Oscillator[19]

Crystal Oscillator in our project

The Microcontroller that we are using does not have an internal oscillator, so we have used the crystal oscillator of rating 40.00 MHz. The PIC microcontroller actually requires 4 crystal periods for executing every instructions. So the real clock speed of our clock will be 10 MHz.

3.39 : Internal Circuitry and working principle of Crystal Oscillator:

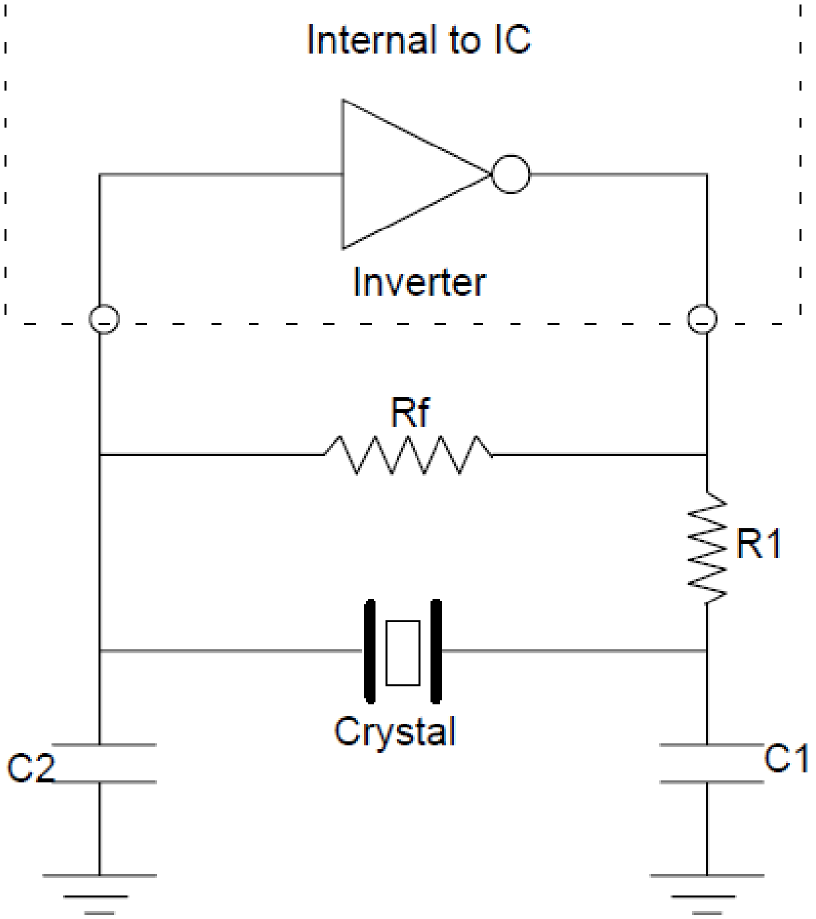


Fig 3.39: Internal circuit diagram of a crystal oscillator[20]

From the figure 3.39, we observe the recommended Crystal oscillator circuit diagram. In this type of setup the crystal is expected to oscillate in parallel resonant mode. The inverter which is internal to the chip acts as class AB amplifier and provides approximately 180° phase shift from input to the output and the p network formed by the crystal, R1, C1 and C2 provides additional 180° phase shift. So the total phase shift around the loop is 360°. This satisfies one of the conditions required to sustain oscillation. The other condition, for proper startup and sustaining oscillation is the closed loop gain should be ^{3}1.The resistor Rf around the inverter provides negative feedback and sets the bias point of the inverter near mid-supply operating the inverter in the high gain linear region. The value of this resistor is high, usually in the range of a 500KW ~2MW[21]

Chapter 4

Schematic Overview:

Our Schematic of our project is divided into 3 parts

1. **The Microcontroller circuit**
2. **PWM Inversion circuit**
3. **The Motor Controller circuit**

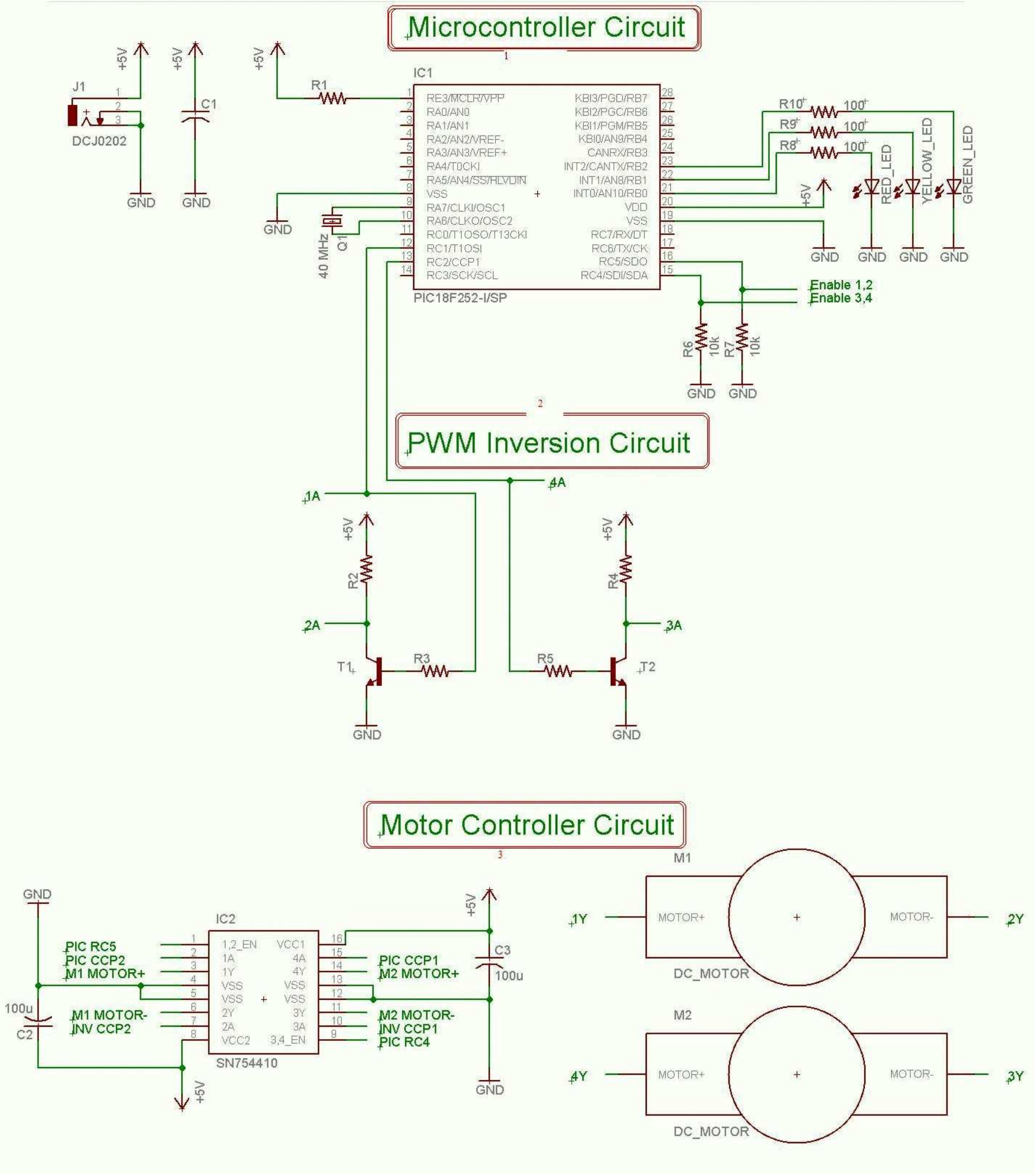


Fig 4.1: The Schematic overview

4.1 **The Microcontroller Circuit**

Seen at the very top rectangular marked with 1, the microcontroller circuit consists of the PIC 18F252, its 40 MHz crystal, a few LEDs, the enable lines going to the motor controller and the PWM signals going to the PWM inversion circuit and the motor controller.

4.2 **The PWM Inversion Circuit**

Two transistors (2n2222A) are used to invert the two PWM signals that are travelling to the motor controller, this part is marked with number 2 in the fig.4.1 . This is done because the motor controller needs to have both the 'positive' PWM signal and the 'negative' PWM signal in what is called a differential pair. The motor controller cannot control our DC motors without both the positive and negative PWM signals.

4.3 **The Motor Control Circuit**

The main part used for the motor control circuit is the L293D. This is a quadruple half h-bridge motor controller IC, this part is ,marked with number 3 in the fig. 4.1. To use this motor controller, it is must to input a differential PWM signal, provide power and ground and enable the bank to start-up the motors. The control signals that will tell the motor controller to do all this will be sent from the PIC Microcontroller.

4.31:Theory for motor control circuit:

Before proceeding to the theory, let's have quick look at PWM Signal

PWM: Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width. To get varying analog values, you change, or modulate, that pulse width.

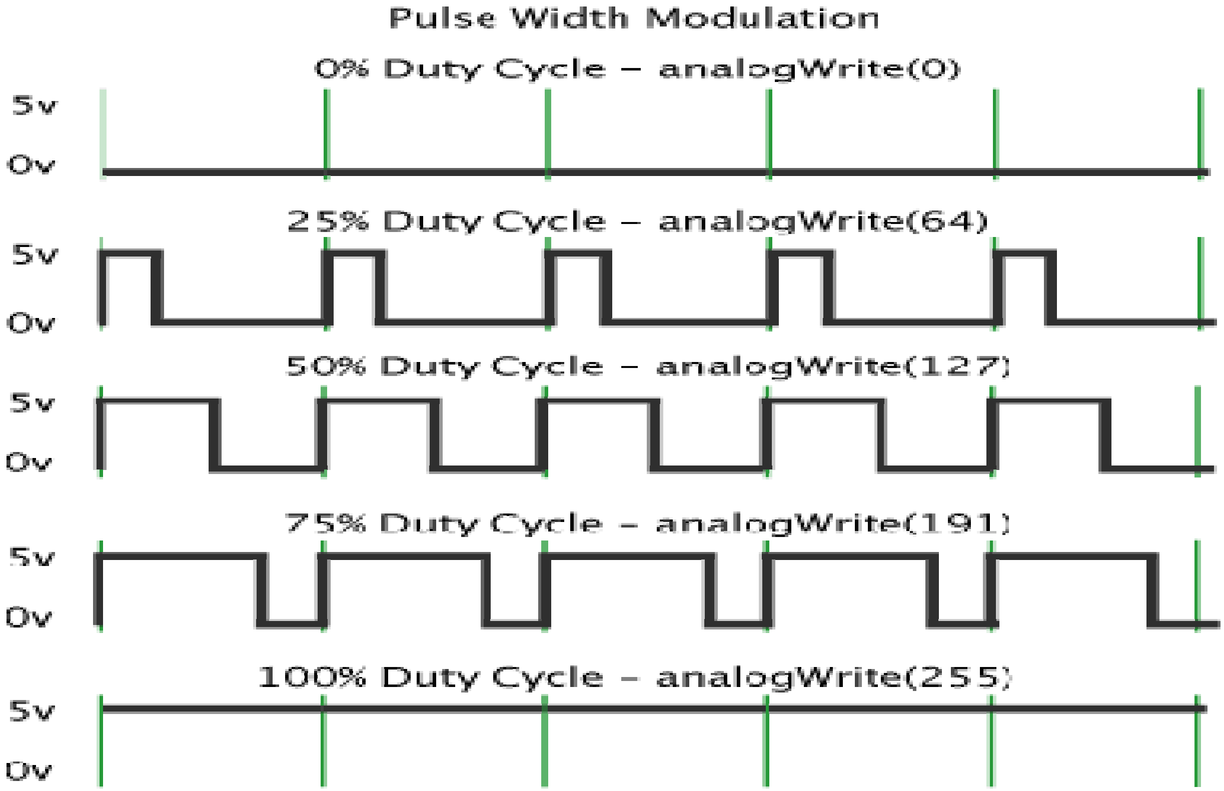

Pulse Width Modulation
0% Duty Cycle – analogWrite(0)
5v
0v
25% Duty Cycle – analogWrite(64)
5v
0v
50% Duty Cycle – analogWrite(127)
5v
0v
75% Duty Cycle – analogWrite(191)
5v
0v
100% Duty Cycle – analogWrite(255)
5v
0v

Fig 4.31: Different types of PWM pulses[22]

In the Fig4.31a above, the green lines represent a regular time period. This duration or period is the inverse of the PWM frequency. In other words, with PWM frequency at about 500Hz, the green lines would measure 2 milliseconds each. A call to analog Write() is on a scale of 0 - 255, such that analog Write(255) requests a 100% duty cycle (always on), and analog Write(127) is a 50% duty cycle (on half the time).

The Theory:

We will describe the theory how the motor control system works in two parts. In the first part we'll describe the input signal called PWM short for Pulse with Modulation and the second will describe which pulses go where to control speed and direction.

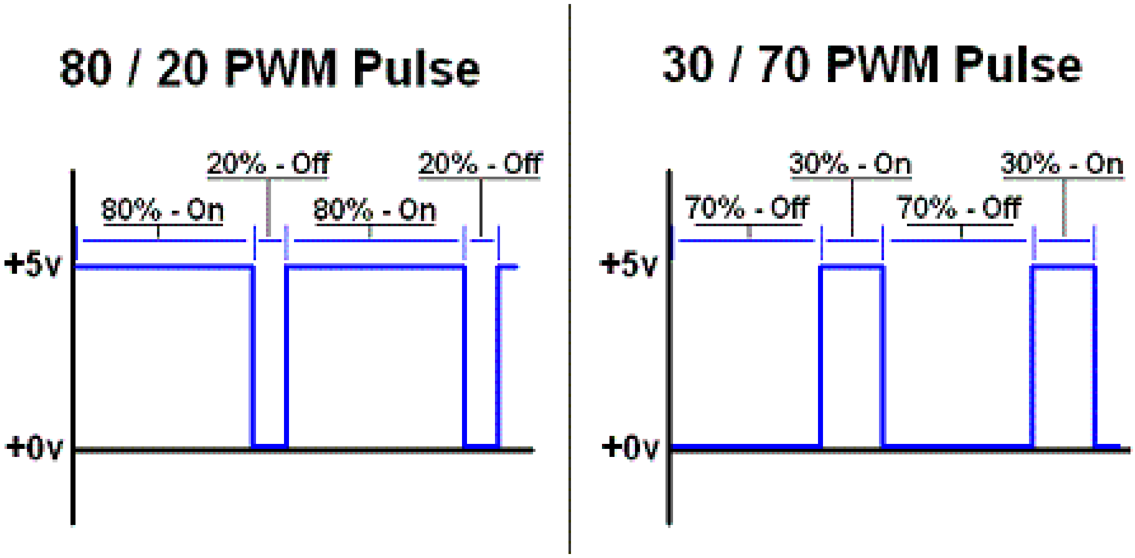


Fig 4.32: Two PWM signal for controlling our motors

The Fig 4.32b above show us two different PWM signals. The first signal is "on" 80% of the time. That means it is at +5v for 80% of the period of the signal. This is a powerful signal and if input into the L293D, would make a motor spin very fast forward. The second image shows you a signal similar to the first, but it is only on 30% of the time. It is at +0v for 70% of the period of the signal. Since this PWM signal is in an 'off' state for the majority of the time, this will actually make the L293D tell the motor to spin in reverse at a medium speed. The center point, 50%/50% tells the motors to stay idle with no movement.

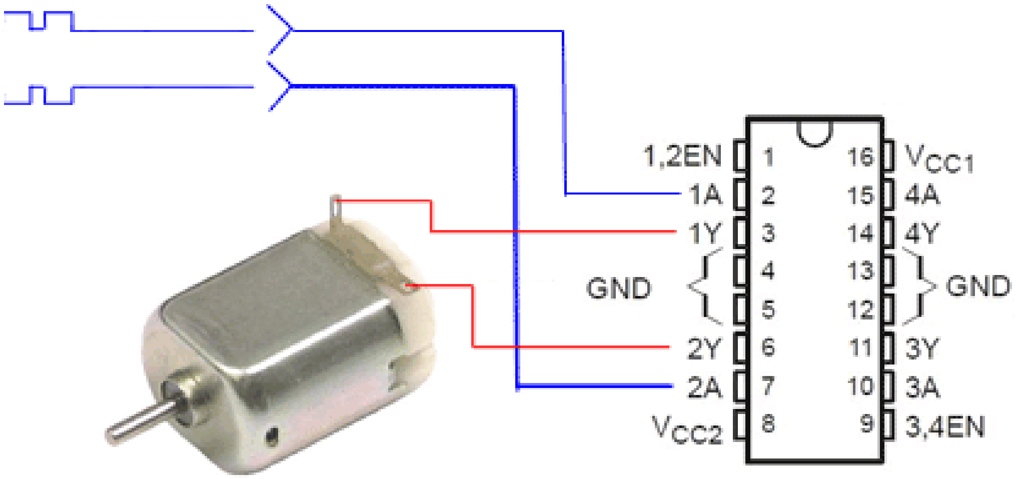


Fig 4.33: Positive PWM input to a L293D motor driver

Looking at one side of the SN754410, we have a motor connected to 1Y and 2Y. A PWM signal that is **80% on // 20% off** is input into 1A and the differential signal to 2A. The corresponding output to the motor makes it turn very fast (almost full speed) in the clockwise direction.

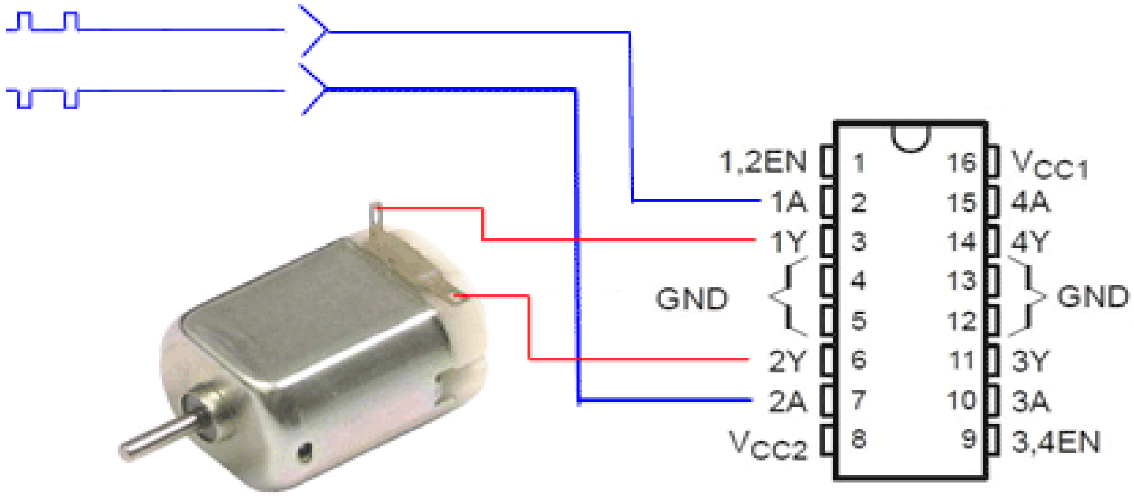


Fig 4.34: Negative PWM input to a L293D motor driver

Similarly, If we look at the same setup but change the input PWM to **20% on // 80% off** , the motor will spin counter-clockwise (reverse) rather at roughly the same speed as the previous example. The motor spins in the opposite direction because the current is being input on the other side of the motor (pin 2A), thusly it flows the opposite direction through the motor compared to the first example.

CHAPTER 5

The Software

There are two main portions of code that we are concerned with:

The Initializations
Motor Control Steps

The first part of the firmware program that we'll look at is the initializations. These configure the PIC's outputs and setup the PWM hardware module. PORTB and PORTC will be used as outputs, so TRISC and TRISB are cleared. Timer2 is used with the hardware PWM module, so we need to set it's pre-scalar value. Finally, the PWM modules are opened up and set to about 7.5 KHz.

5.1The Initializations

====Beginning of code====

```
TRISB = 0x00;

PORTB= 0x01; //Red LED ON


TRIC=0x00;

PORTC=0x00;

Delay 10kTCYx(1);

//use x16 pre scalar

2CONbits.T2CKPS0 = 1;

T2CONbits.T2CKPS1 = 1;
```

```
//10 KHz
OpenPWM1(20);
OpenPWM2(20);

//Disable Both Motors
PORTC = 0x00;
=====End of Code====
```

5.2 **The Motor Control Steps**

This part of code from the firmware program gives an example for how to control the PWM output on the PIC. Both PWM modules need to be changed at the same time, to keep the motors in-sync. The first example shows how the motors were turned-on and told to move full speed forward (90% Duty Cycle). The second example shows how the motors are put into a stop state (50% duty cycle + disable motors).

```
====Beginning of code===
##Move Forward
SetDCPWM1(6);
SetDCPWM2(6);
PORTB = 0x04; //GREEN LED ON
PORTCbits.RC4 = 1;
PORTCbits.RC5 = 1;


//1 Second Delay
Delay10KTCYx(250);
Delay10KTCYx(250)


##Full Stop
SetDCPWM1(38);
SetDCPWM2(38);
PORTB = 0x01; //RED LED ON
PORTCbits.RC4 = 0;
PORTCbits.RC5 = 0;


//1 Second Delay
Delay10KTCYx(250);
Delay10KTCYx(250);
```

=====End of Code====

5.3 **The Main Software**

This code is written by C programming language, compiled in PIC Platfrom of Code Blocks Software and simulated in Proteus ISIS professional 7.

This firmware program is meant for the PIC 18F252. It controls two motors using the PWM modules and it sets three LEDs on PORTB.

Delays are used to tell the motors to move a few specific moves.

```
====Beginning of Code===
#include <p18f252.h>
#include <pwm.h>
#include <delays.h>

#pragma config OSC  = HS
#pragma config WDT = OFF
#pragma config LVP   = OFF

void set_motor_speed(unsigned int,unsigned int);

void main(void){

TRISB = 0x00;
PORTB = 0x01; //RED LED ON
```

```
TRISC = 0x00;
PORTC = 0x00;

Delay10KTCYx(1);

//Use x16 Pre-scalar
T2CONbits.T2CKPS0 = 1;
T2CONbits.T2CKPS1 = 1;

// PWM Period: 10 KHz
OpenPWM1(20);
OpenPWM2(20);

/*
To control motor speed and direction use the
SetDCPWM1(x);
SecDCPWM2(x);
functions, where x is a value between 6 and 70.
The short table below explains how the numbers
correlate to speed/direction:
 6 = 90% - Full Forward
 38 = 50% - Motors OFF
70 = 10% - Full Reverse
```

```
*/


//Disable Both Motors
PORTC = 0x00;


//Sequence:
// Move Forward
// Turn-Around 180 Degrees
// Move Forward (to original position)
// Turn-Around 180 Degrees


while(1){
/*******Full Stop***********/
SetDCPWM1(38);
SetDCPWM2(38);
PORTB = 0x01; //RED LED ON
PORTCbits.RC4 = 0;
PORTCbits.RC5 = 0;


//1 Second Delay
Delay10KTCYx(250);
Delay10KTCYx(250);
Delay10KTCYx(250);
```

```
Delay10KTCYx(250);
/***************************/


/*******Move Forward***********/
SetDCPWM1(6);
SetDCPWM2(6);
PORTB = 0x04; //GREEN LED ON
PORTCbits.RC4 = 1;
PORTCbits.RC5 = 1;


//1 Second Delay
Delay10KTCYx(250);
Delay10KTCYx(250);
/***************************/


/*******Full Stop***********/
SetDCPWM1(38);
SetDCPWM2(38);
PORTB = 0x01; //RED LED ON
PORTCbits.RC4 = 0;
PORTCbits.RC5 = 0;


//1 Second Delay
```

```
Delay10KTCYx(250);

Delay10KTCYx(250);

/***************************/


/*******Turn Around***********/

SetDCPWM1(6);

SetDCPWM2(70);

PORTB = 0x02; //YELLOW LED ON

PORTCbits.RC4 = 1;

PORTCbits.RC5 = 1;


//0.30 Second Delay

Delay10KTCYx(250);

/***************************/


/*******Full Stop***********/

SetDCPWM1(38);

SetDCPWM2(38);

PORTB = 0x01; //RED LED ON

PORTCbits.RC4 = 0;

PORTCbits.RC5 = 0;


//1 Second Delay
```

```
Delay10KTCYx(250);
Delay10KTCYx(250);
/******************************/

/*******Move Forward***********/
SetDCPWM1(6);
SetDCPWM2(6);
PORTB = 0x04; //GREEN LED ON
PORTCbits.RC4 = 1;
PORTCbits.RC5 = 1;

//1 Second Delay
Delay10KTCYx(250);
Delay10KTCYx(250);
/******************************/

/*******Full Stop***********/
SetDCPWM1(38);
SetDCPWM2(38);
PORTB = 0x01; //RED LED ON
PORTCbits.RC4 = 0;
PORTCbits.RC5 = 0;
```

```
//1 Second Delay

Delay10KTCYx(250);

Delay10KTCYx(250);

/***************************/


/*******Turn Around***********/

SetDCPWM1(6);

SetDCPWM2(70);

PORTB = 0x02; //YELLOW LED ON

PORTCbits.RC4 = 1;

PORTCbits.RC5 = 1;


//0.30 Second Delay

Delay10KTCYx(250);

/***************************/


/*******Full Stop***********/

SetDCPWM1(38);

SetDCPWM2(38);

PORTB = 0x01; //RED LED ON

PORTCbits.RC4 = 0;

PORTCbits.RC5 = 0;
```

```
//1 Second Delay
Delay10KTCYx(250);
Delay10KTCYx(250);
/***************************/

/*******Move Forward***********/
SetDCPWM1(6);
SetDCPWM2(6);
PORTB = 0x04; //GREEN LED ON
PORTCbits.RC4 = 1;
PORTCbits.RC5 = 1;

//1 Second Delay
Delay10KTCYx(250);
Delay10KTCYx(250);
/***************************/

/*******Full Stop***********/
SetDCPWM1(38);
SetDCPWM2(38);
PORTB = 0x01; //RED LED ON
PORTCbits.RC4 = 0;
PORTCbits.RC5 = 0;
```

```
//1 Second Delay
Delay10KTCYx(250);
Delay10KTCYx(250);
/***************************/

/*******Move Backward***********/
SetDCPWM1(70);
SetDCPWM2(70);
PORTB = 0x04; //GREEN LED ON
PORTCbits.RC4 = 1;
PORTCbits.RC5 = 1;

//1 Second Delay
Delay10KTCYx(250);
Delay10KTCYx(250);
Delay10KTCYx(250);
Delay10KTCYx(250);
/***************************/

/*******All Stop***********/
SetDCPWM1(38);
SetDCPWM2(38);
```

```
PORTB = 0x01; //RED LED ON
PORTCbits.RC4 = 0;
PORTCbits.RC5 = 0;

//Pause Forever
while(1);
/***************************/

}



ClosePWM1();
ClosePWM2();



}
```

CHAPTER 6

Testing & Simulations:

6.1 About testing and simulations

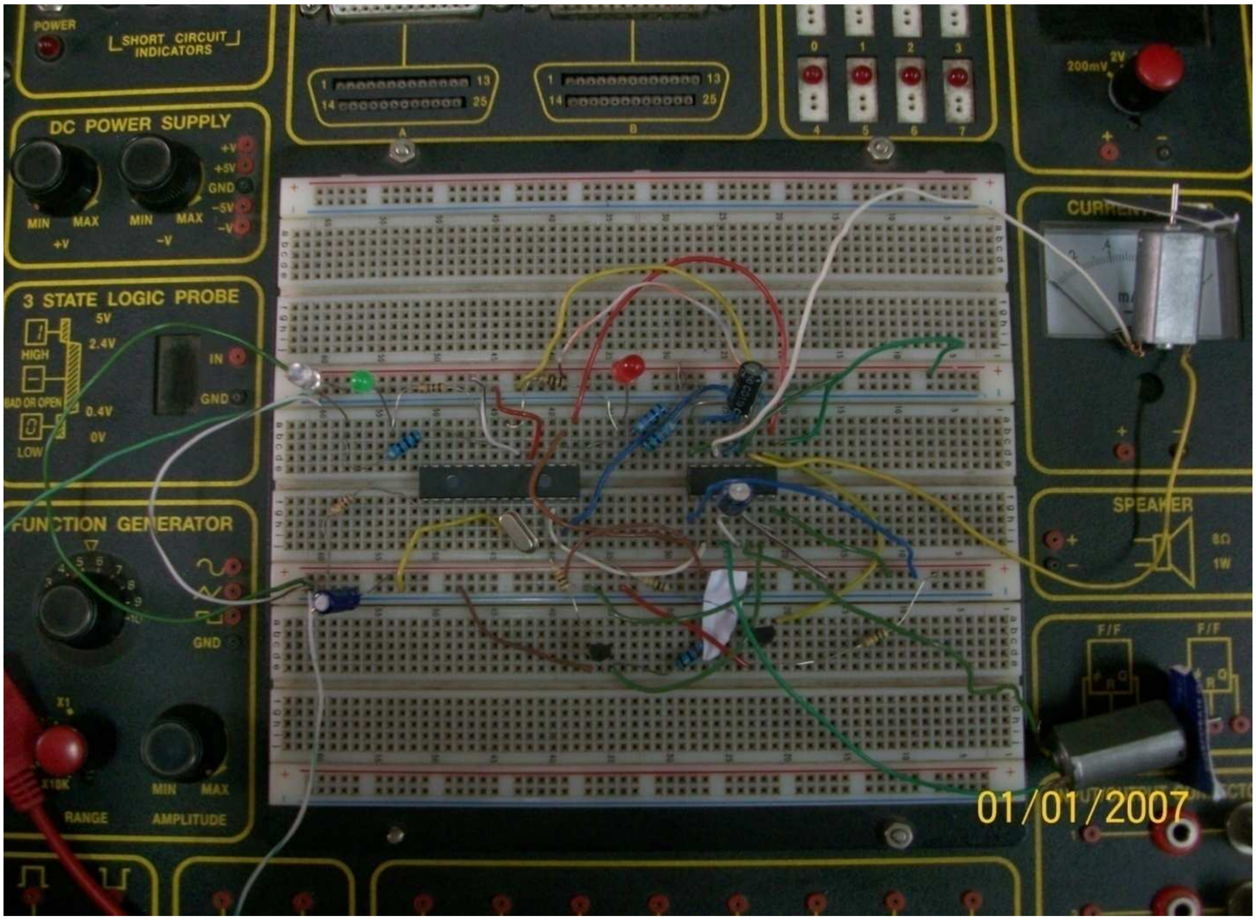
The instructions we wrote in the microcontroller, is to:

1. First turn on the Red LED
2. Rotate the motor in forward or positive direction, for moving the car forward
3. Stop Everything
4. Rotate one motor in positive direction and other in reverse direction, for turning the car
5. Turn on the Green LED
6. Rotate the motors in forward direction, for moving the car forward
7. Stop
8. Rotate one motor in positive direction and other in reverse direction, for turning the car
9. Yellow LED ON
10. Rotate the motors in forward direction, for moving the car forward
11. Rotate the motors in reverse direction, for moving the car back.

We've tested every of these instructions and they were fully functioned.

In the lab we’ve performed various test & simulation for many days, and got the desired result. We’ve tested for cuircuitry, standard PWM signal, reverse PWM signal, voltage level, current level, motor’s polarity, etc.

We’ve captured some of the testing’s in the LAB.



01/01/2007

Fig 6.1: The Circuitry test

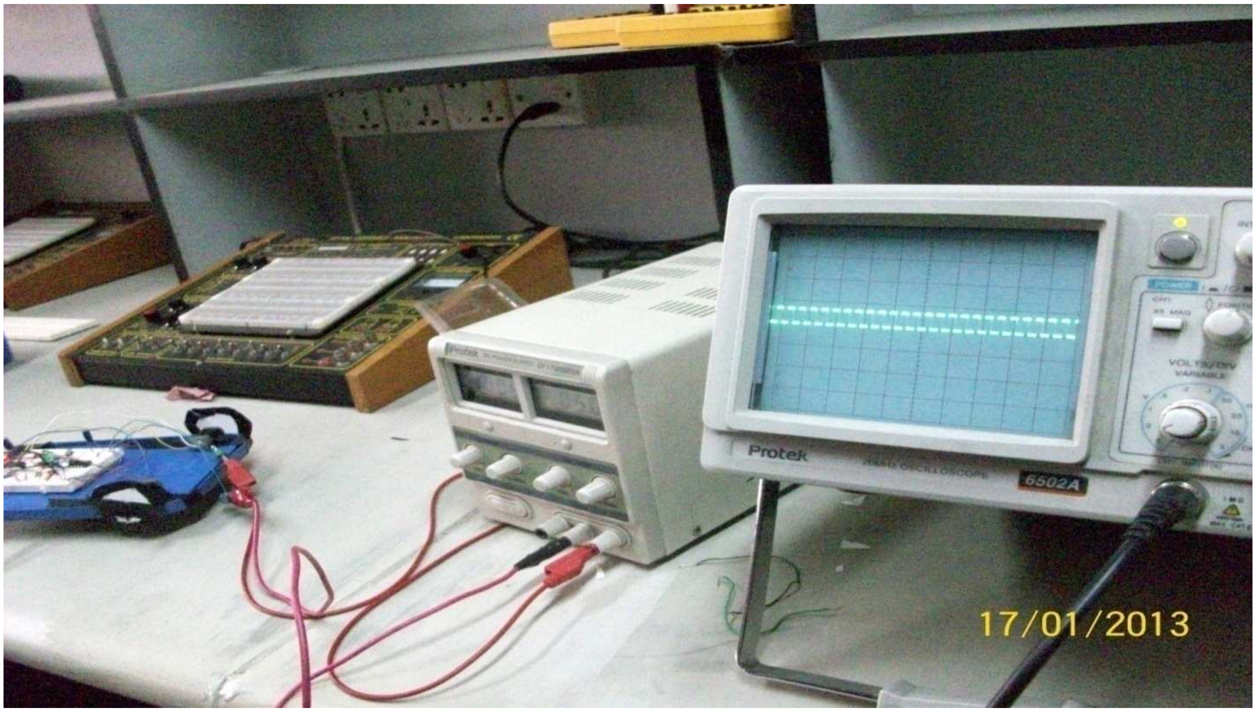


Fig 6.2: The Input/output voltage & Current level test

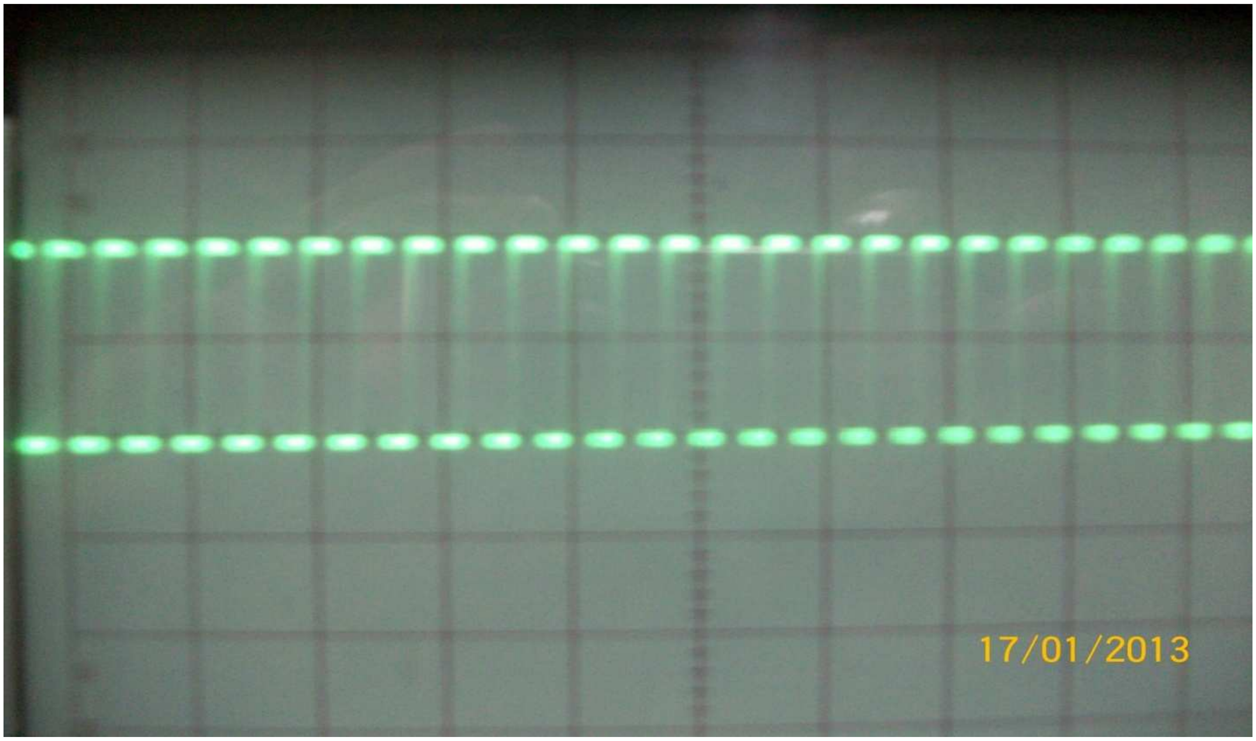


Fig 6.3: The Standard PWM wave-shape test (Oscilloscope monitor)

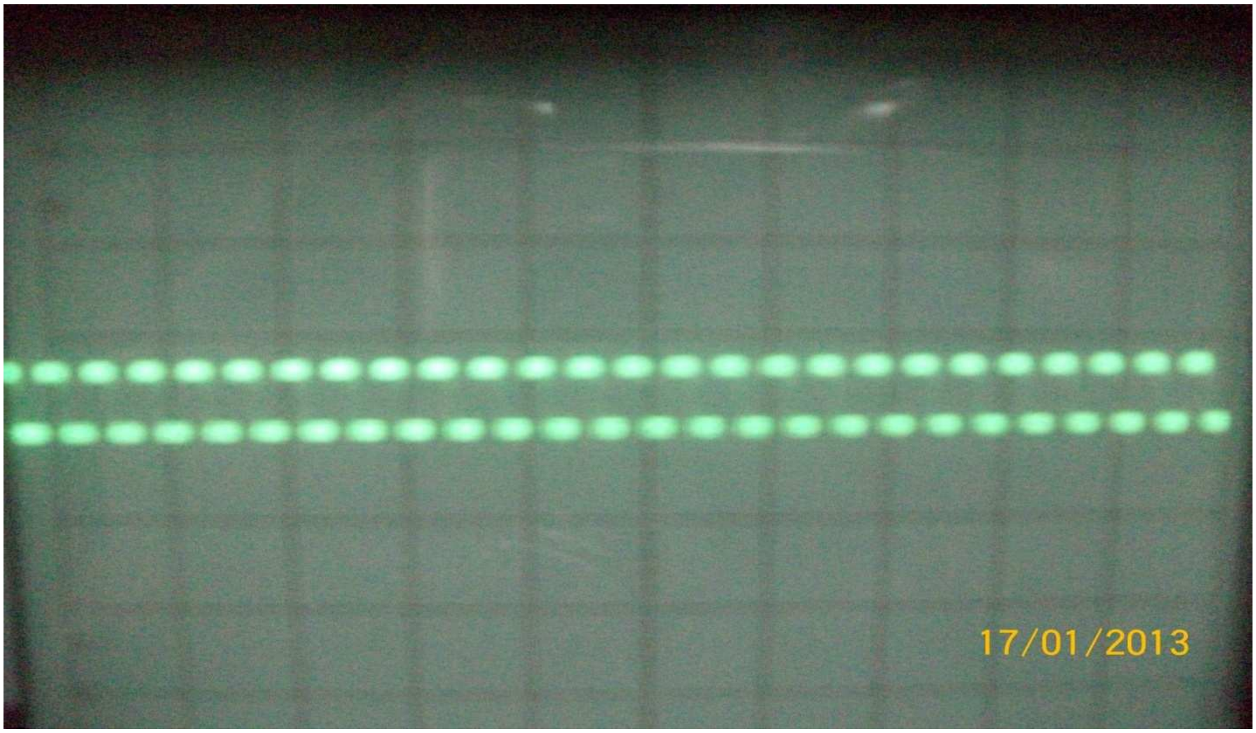


Fig 6.4: Output from the transistor’s Collector

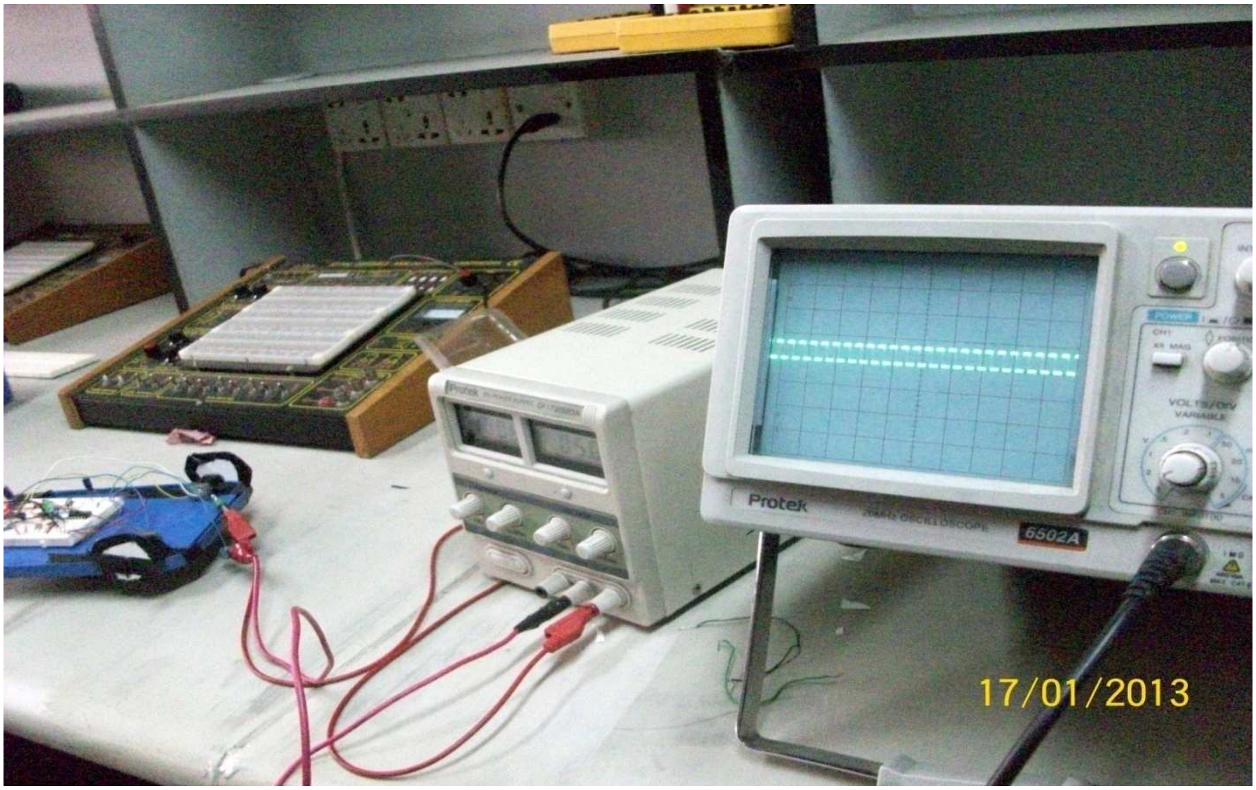


Fig 6.5: Inverse PWM in the motor input

Chapter 7

Assembling & Chassis

7.1 About Assembling & Chassis

For assembling the whole circuit, we’ve cut a normal bread board into two pieces, and assembled all the parts in a half sized bread board. It was quiet difficult task for us to making the chassis as it must be light & plain cause our motors are not capable of carrying heavy weight, and it has to be plain for setup the bread board, and other parts. After choosing many materials for this purpose, the next task for us were to selecting the most appropriate one, at last we’ve selected a ice cream box for making the chassis, as it is light in weight and plain in nature. The wheel were from the toy cars, and from that time the making of the real body was a matter of time, and we’ve succeeded to make it as our desired one. This is the real scenario after all the works were done, of course we’ve named this robotic car as “BIJOY” in Bengali “বিজয়”

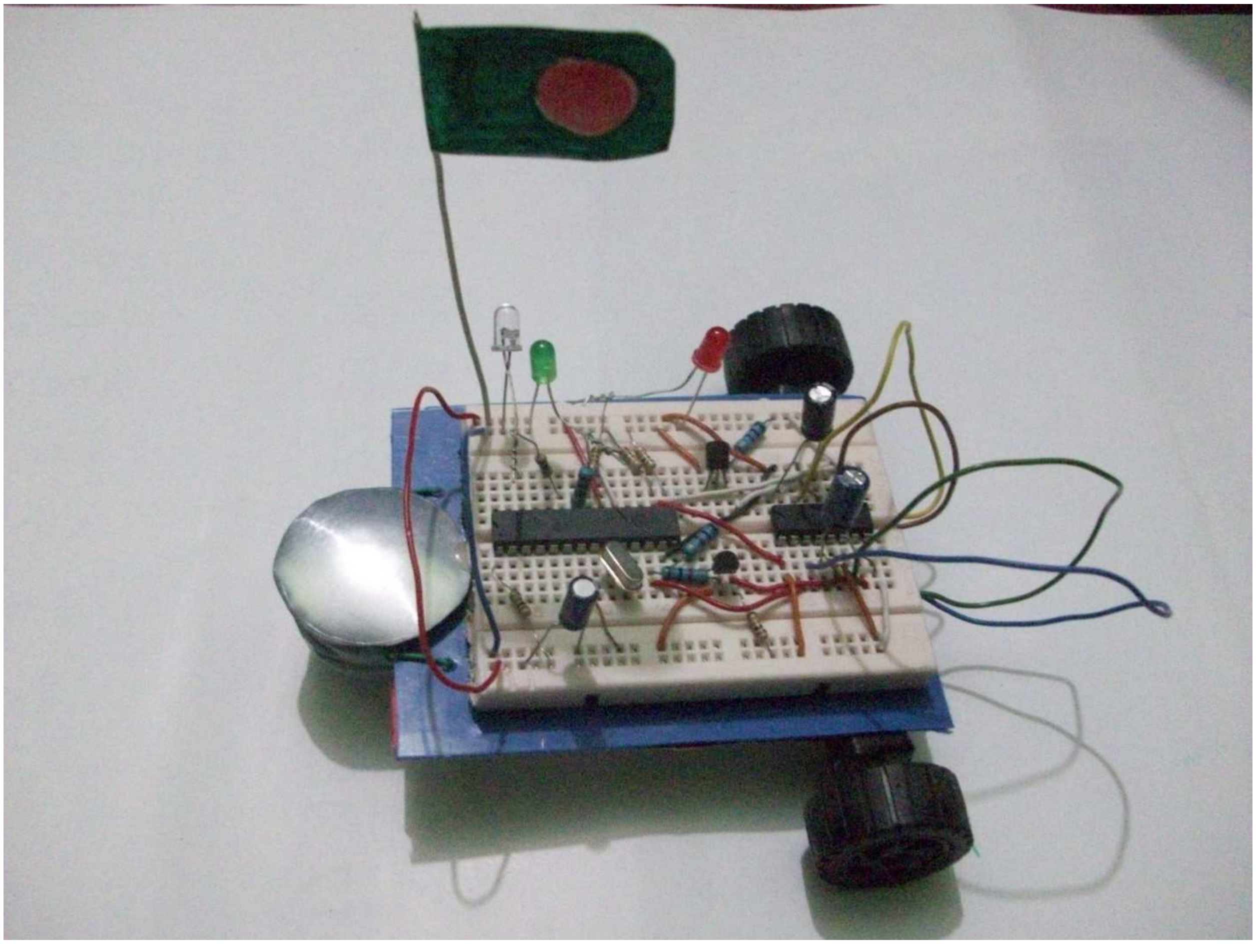


Fig7.1: The BIJOY “বিজয়” at its real look

7.2 **Future Modifications**

In future it is possible to modify this robotic car by giving more instructions. The motors used could be replaced with more strong motors, for carrying heavy load. This car further can work as a security robot, if a video camera could be attached. If sensors, like the land mine sensor, metal detector sensor, gas detector sensor could be attached it can function like a complete security module. If a GPRS device could be attached, it could execute the program given by a http server in real time.

Bibliography

[1] http://www.embeddedsystem.com/

[2] The 8051 microcontroller and Embedded systems – Muhammad Ali Mazidi, Janice Gillispie Mazidi

[3] http://www.webopedia.com/TERM/E/embedded_system.html

[4] http://en.wikipedia.org/wiki/Embedded_system

[5] http://www.embeddedsystem.com/

[6] http://produto.mercadolivre.com.br/MLB-457050471-10-microcontrolador-pic18f252-ip-pic-18f252-_JM?redirectedFromParent=MLB446427108

[7] Microchip PIC18FXX2 Data Sheet

[8] Microchip PIC18FXX2 Data Sheet

[9] http://www.futurlec.com/Microchip/PIC18F252.shtml

[10] SGS-THOMSON MICROELECTROCICS L293D.pdf

[11] SGS-THOMSON MICROELECTROCICS L293D.pdf

[12] http://ahmadsaffuanzainuddin.blogspot.com/2012/11/circuit-explanation-l293d-motor-driver.html

[13] http://www.electroons.com/8051/electroons/dc_motor_control.html

[14] http://www.st.com/2N2222A.pdf

[15] http://www.ebay.com/itm/50-2N2222A-NPN-Transistors-2N-2222A-2N2222-TO-92-/150518052914

[16] http://www.microsemi.com/Datasheet# MSC0275A

[17] http://www.st.com/2N2222A.pdf

[18] http://en.wikipedia.org/wiki/Crystal_oscillator

[19] http://szcanyi.en.alibaba.com/product/308300831-200036998/18_432MHZ_HC49S_dip_crystal_Oscillators.html

[20] MX.COM,INC.Mixed Signal ICs Crystal Oscillator Circuit Design

[21] MX.COM,INC.Mixed Signal ICs Crystal Oscillator Circuit Design

[22] http://arduino.cc/en/Tutorial/PWM